Scripts for Running CERES PGE's on SGE

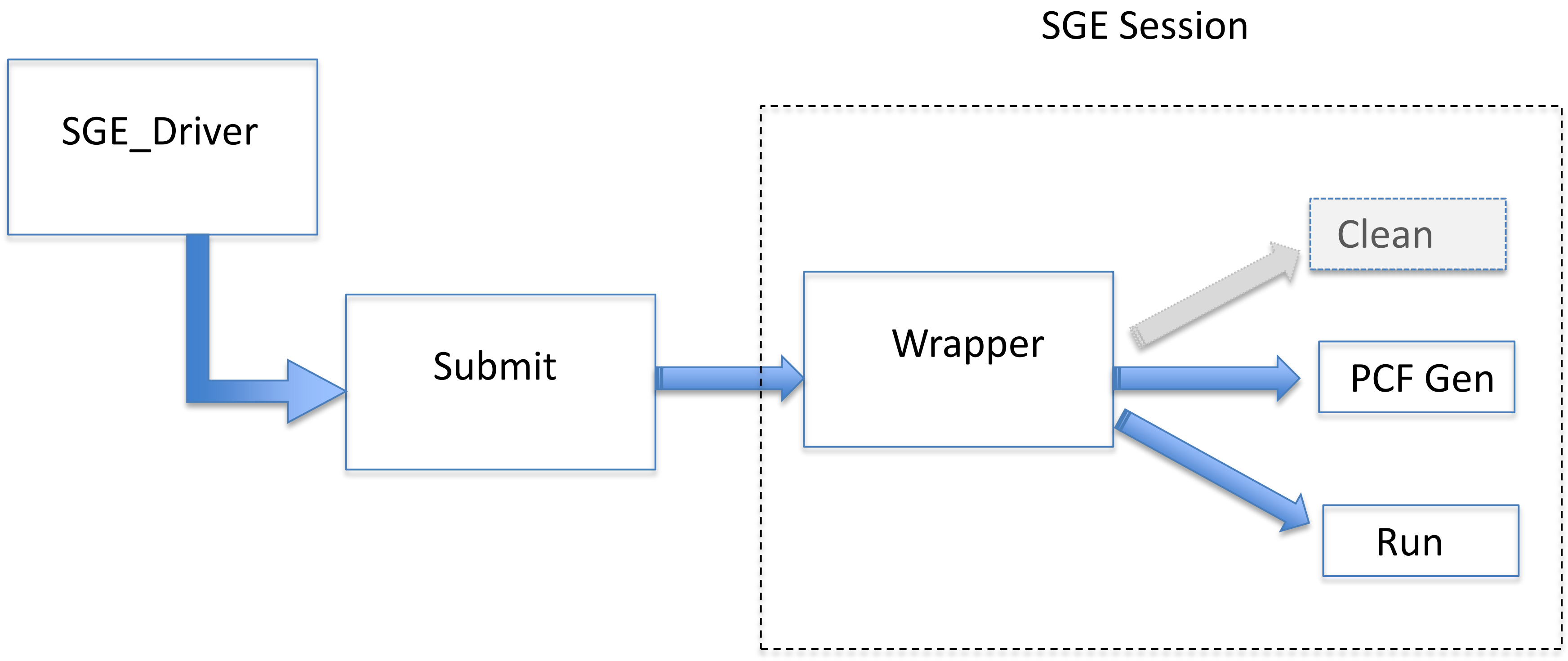
Brian Magill

5/10/2010

Introduction

- *Need to run PGE’s on SGE independent of platform submitted to*
- *Develop scripts to setup for either environment on the fly*
- *Currently in prototype phase*
  - *Development process tried on several PGE’s to get a feel*
  - *Currently Focused on Instrument Subsystem Delivery*
  - *Initial PGE sent to SSIT is CER1.1P7*
- *Design by Scott Zentz*
- *Documentation by Brian Magill*

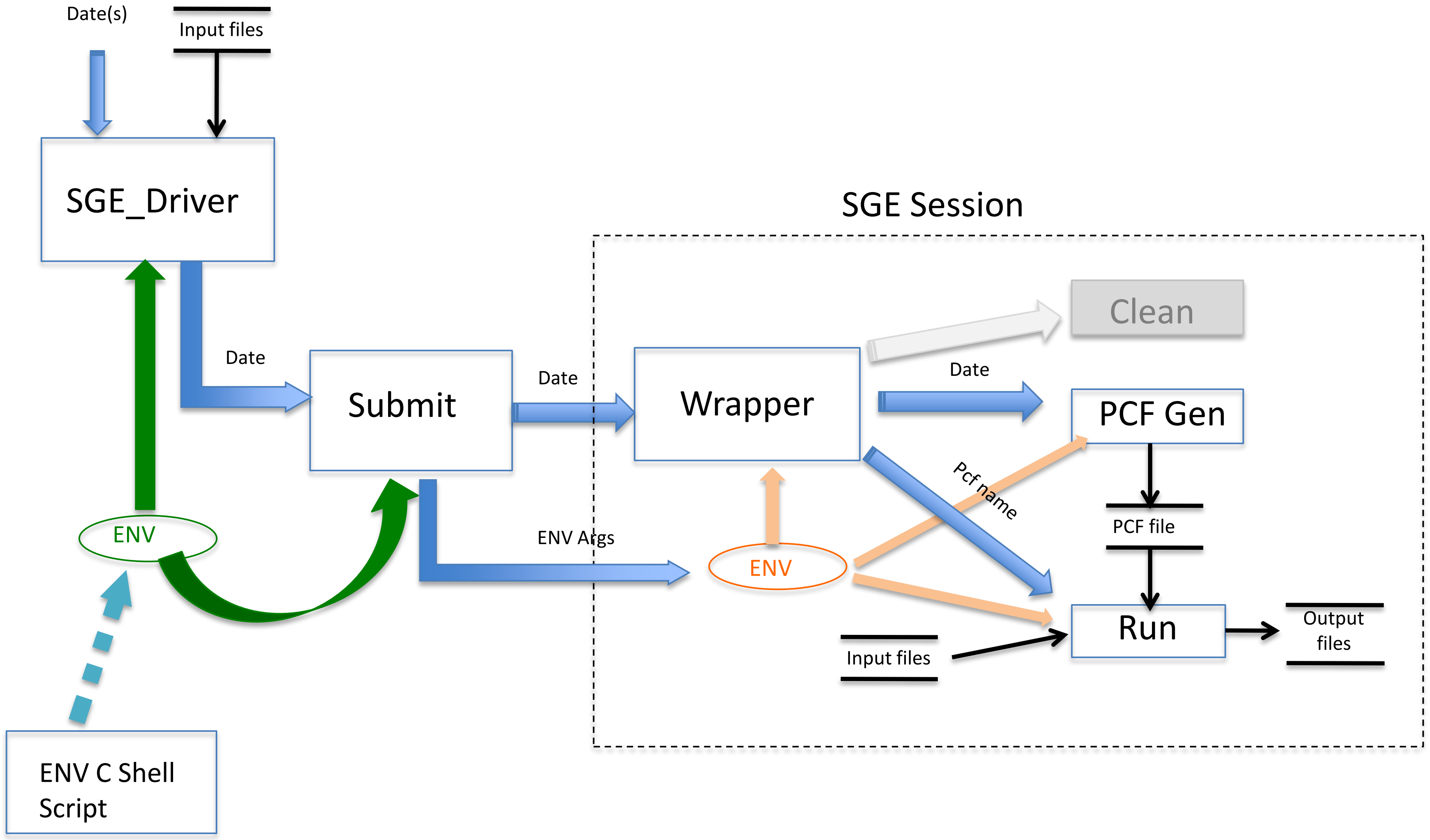
Top Command Flow



Process Flow

- Manually source PGE Environmental C shell script prior to running scripts
- Run SGE_Driver script with date(s) or time(s) of interest
- SGE_Driver checks files and makes one or more system calls to run Submit Script if they are okay.
- Submit Script submits Wrapper script and environment to SGE
- Wrapper Script makes system calls to the PCF generator script, followed by system call to run script for PGE executable. (possibly a clean script, TBD)
- Note: PCF generator and Run scripts can also be run stand alone.

Top Level Data Flow



Top Data Flow

- Aside from Date/Time, all other PGE specific data is passed via environmental variables to SGE.
  - Roughly 15 variables.
  - Convenient way to get information into SGE session.
- These environmental variables are mostly stored in the SubSystem hash table.
- All Perl scripts shown in diagram are currently independent programs
- PCF generator, Run script, SGE all produce log files. SGE_Driver produces an error log if it cannot submit a job.
- Naming convention for scripts: <PGE name>-<function>.pl   *(i.e., CER1.1P7-SGE_Driver.pl, CER1.1P7-Submit.pl, etc.)*

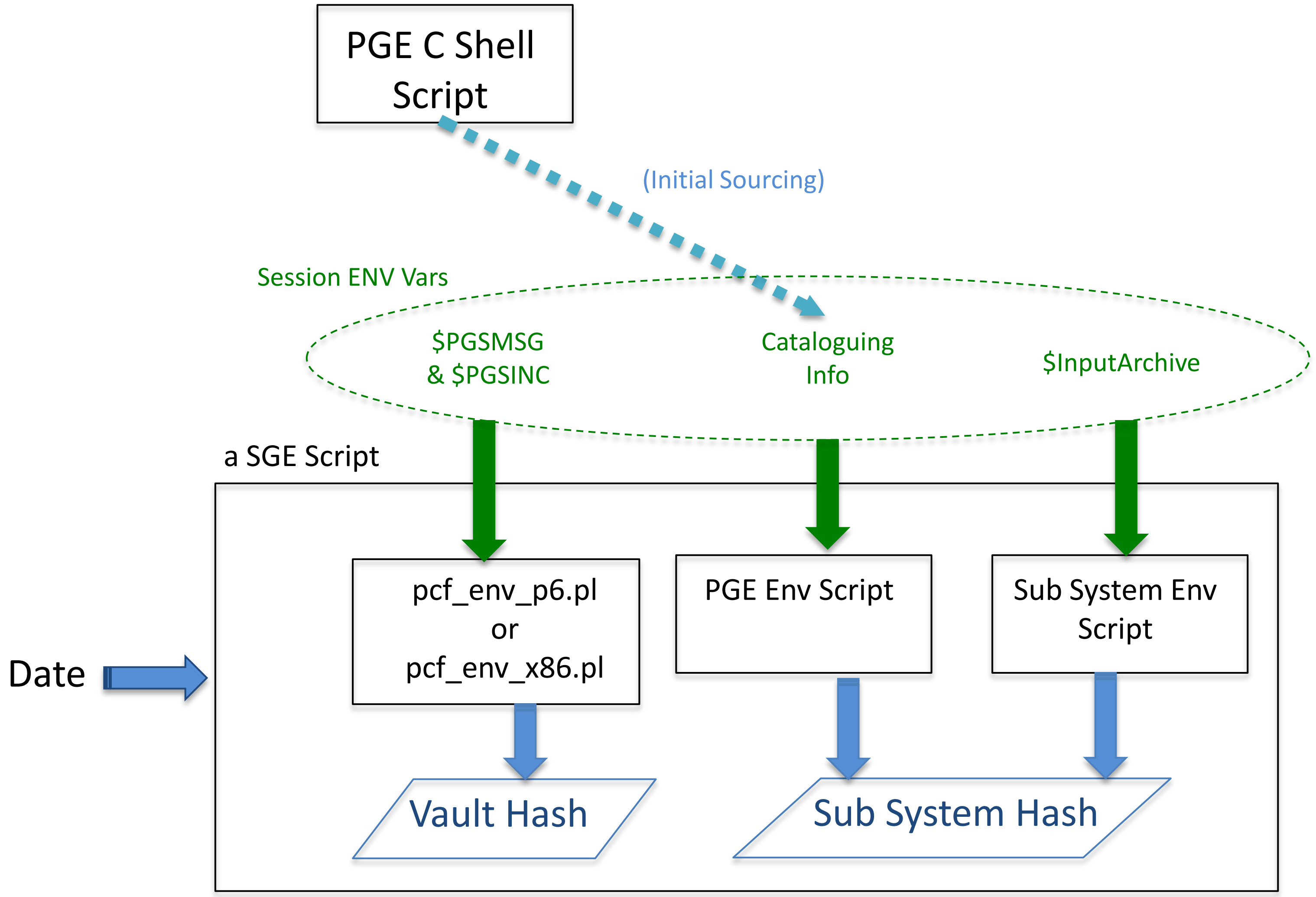
SGE Scripts Extensively Use Hash Tables

- Data structure associating keys with values
- Also known as an associative array, dictionary, or map depending upon computer language.
- Like an array, but “index” is a string instead of an integer
- Stored unordered. Not guaranteed to be contiguous in memory

```
# Perl example:

my %hash;
$hash{‘PS1_1’} = ‘Edition1-CV’;
print $ENV{‘HOSTTYPE’}, “\n”;
```

Getting Processing Info into SGE Scripts



Hash Tables common to scripts

Vault Hash
CPU Brand
CERESLib Directory
Toolkit Directories
Load Libraries

Subsystem Hash
PGE Name
Directory Paths
Sampling Strategy Components
Production Strategy Components
Configuration Code

Cataloguing Info

Vault Hash Table

- Platform specific (X86 or P6)
- Setup in pcf_env_p6.pl and pcf_env_x86.pl
- Specifies CERESLIB and Toolkit Directory names
- Values hardcoded in pcf_env scripts except for PGS message and include directory names

Subsystem Hash Table

- Contains sections for subsystem wide and PGE specific variables.
- Subsystem section is set in subsystem environmental Perl script
  - Located in subsystem rcf directory
  - Sets variables for subsystem wide directories based upon location of PGE's rcf directory
  - May use environmental variables for input data directory
- PGE section is set in PGE environmental Perl script
  - Located in each PGE's rcf directory
  - Sets variable for PGE name
  - Sets variables for PGE specific directories based upon home directory location
  - Sets variables for cataloguing: sampling & production strategies and configuration code
    - Initially these values are read from environmental variables set by PGE C shell script.
    - All environmental variables set in this C shell script must be accounted for in hash table!

Other Scripts Affected by Coding Changes

- PCF Generator script
  - Should use vault and subsystem hash instead of %ENV
- Run Script
  - Should use vault and subsystem hash instead of %ENV
- Make scripts
  - Need to be consistent with Toolkit values in Vault Hash
- Clean Scripts (TBD)
- Test Suite
  - Test plans should reflect changes in how files are produced
  - Needs to be able to manipulate and evaluate files produced by SGE scripts

Additional Log File for SGE Session

- Contains everything printed to STDOUT and STDERR during SGE session
- Also includes a header line with job ID and log file name.
- Located in sge_logs/ directory at subsystem level
- Output from Toolkit is still written to runtime logs

SGE_Driver Error Log

- Similar to PCF Log File
- Lists existing and missing files and run time parameters
- Only generated if mandatory files are missing

Example: CER1.4P1 (BDSI Subsetter)

- Creates BDSI files from internal calibration events in BDS files
- C++ replacement for CER1.3P1 (Ada version)
- Environmental Script for FM1: temp-fm1-env.csh
- Driver Script: CER1.4P1-SGE_Driver.pl
- Run directory: ../instrument/CER1.4P1/rcf
- Run logs located in: .. /instrument/runlogs/
- SGE logs located in: ../instrument/sge_logs/
- Data located under: ../instrument/data/

Temp-fm1-env.csh Contents

```
#---------------------------------------------------------------------------
#  temp-fm1-env.csh
#
                         .
                         .
                         .
#
#        set up DAAC production environment variables

      setenv SS1              Terra-FM1
      setenv PS1_0            Edition1-CV
      setenv CC1              033033
      setenv CC1_3            000004
      setenv SW1               716
      setenv SW1_3             716
      setenv DATA1             716
      setenv DATA1_3           716
      setenv SAT               Terra
      setenv INST              FM1
      setenv PGSCONSDIR   $CERESLIB/data
                         .
                         .
                         .
```

Sample Run for CER1.4P1

```
instrument1-blue 156% source temp-fm1-env.csh
instrument1-blue 157% ./CER1.4P1-SGE_Driver.pl -d 20020315
Your job 104469 ("CER1.4P1_Terra-FM1_Edition1-CV_033033_20020315") has been submitted
                        .
                        .
                        .
instrument1-blue 160% qacct -j 104469
==============================================================
qname        scf.q
hostname     bb109
group        instrument
owner        bmagill
project      NONE
department   defaultdepartment
jobname      CER1.4P1_Terra-FM1_Edition1-CV_033033_20020315
jobnumber    104469
taskid       undefined
account      sge
priority     0
qsub_time    Fri May  7 14:21:14 2010
start_time   Fri May  7 14:21:19 2010
end_time     Fri May  7 14:23:45 2010
                        .
                        .
                        .
```

SGE Log File:
CER1.4P1_Terra-FM1_Edition1-CV_033033_20020315.o104469

```
Your job 104469 ("CER1.4P1_Terra-FM1_Edition1-CV_033033_20020315") has been submitted

Compressing BDSI -- CER_BDSI_Terra-FM1_Edition1-CV_000004.2002031501
compressing a Internal_Cal_BDS file
.hdf file to be compressed: /homedir/bmagill/projects/ceres/instrument/data/BDSI/Terra-FM1_Edition1-CV/2002/03/CER_
compressed and reordered .hdf name  /homedir/bmagill/projects/ceres/instrument/data/BDSI/Terra-FM1_Edition1-CV/20

CER1.4P1 SUCCESSFUL -- Exit Status = 0
tar: Removing leading `/' from member names
Instrument Subsystem PGE CER1.4P1 Complete for PCF file CER1.4P1_PCF_Terra-FM1_Edition1-CV_000004.20020315
```

Future Directions

- Override command for job submission
- Replace Need for Environmental C Shell Script
  - Could use Initialization File
  - Ultimately use a database
- Minimized need for Environmental Variables
  - Possibly replace system calls to Perl scripts with subroutine calls?
  - Any alternatives for passing information to Wrapper via SGE?
- Over all program still needs feedback from SSIT and Operations
- Flexibility in approach based upon scientists’ needs

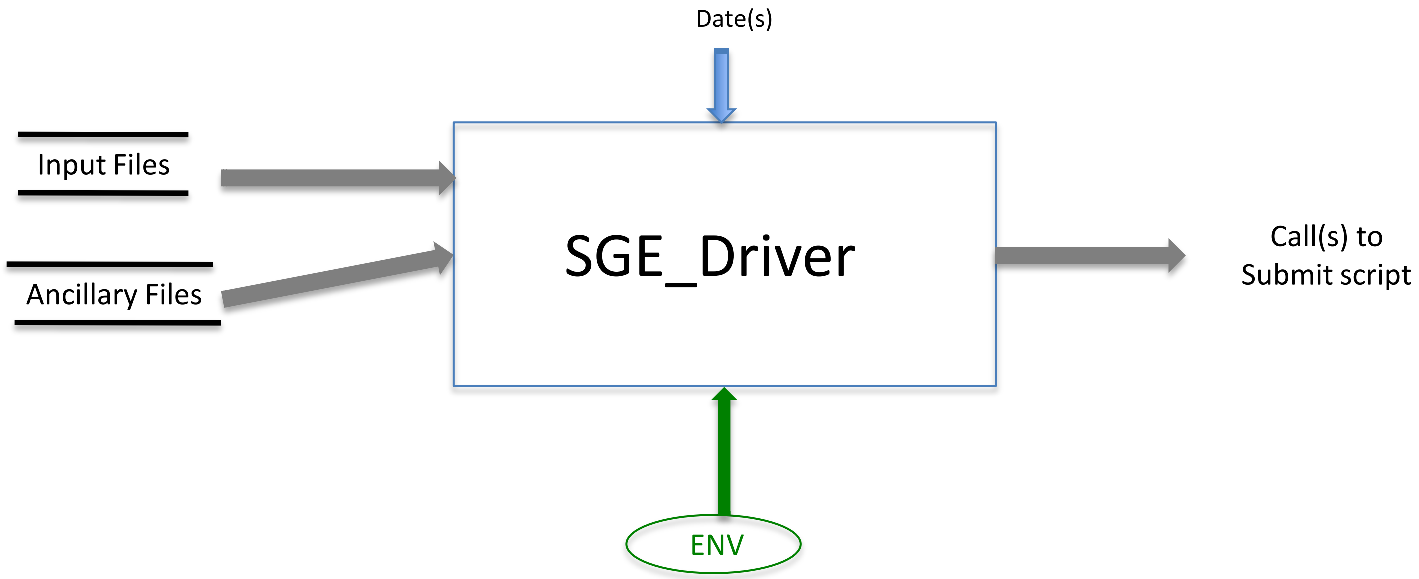
Appendix A: Coding Guidelines

- All Perl scripts should start with “use strict”
- All Perl executable scripts should use the warn option
- Use the Perl interpreter in /usr/local/bin/
- All scripts should have headers with name of script, creation date, author, short description and CVS $Log keyword.
- Values in the subsystem or Vault hash should be used instead of the corresponding %ENV entries. (sometimes the two aren’t the same.)
- Only pass necessary environment variables into SGE session.
- The command line arguments into PGE Driver script should allow for a single date/time or a range of date(s)/time(s)
- SGE script names should be consistent across PGE’s and Sub Systems.

Appendix B: Detailed Data Flow

- Data Flow and Hierarchy Charts for major SGE Scripts

SGE_Driver Script Data Flow

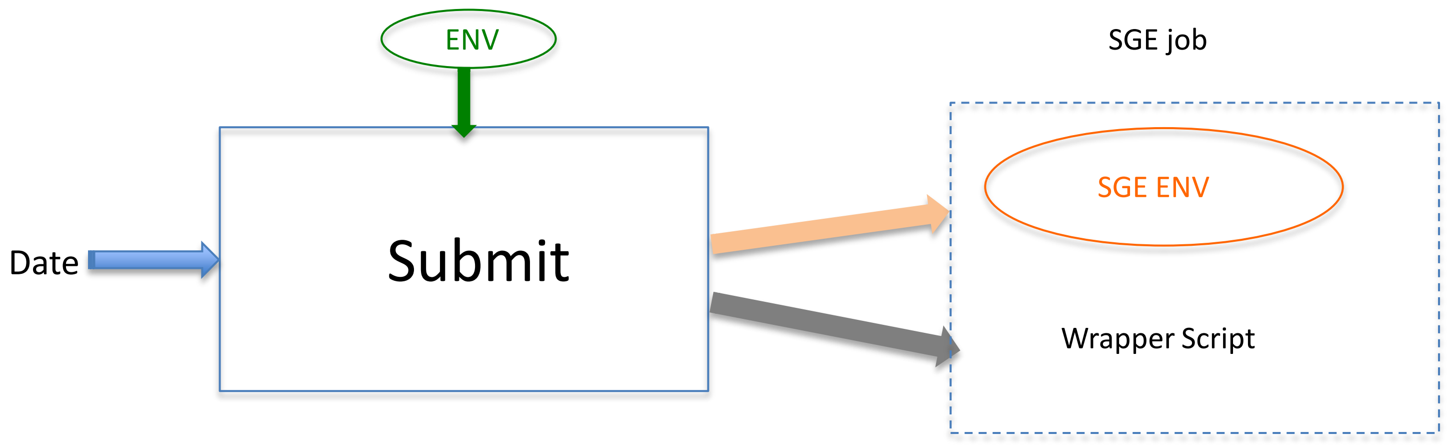


SGE_Driver Pseudo code

```
• Parse command line arguments
• Create list of dates/times (one or more entries)
• Check that all environmental variables in subsystem hash are set
• Loop through list of dates/times:
    – Check input files
    – Check Ancillary files
    – If all of these files are good
        • Submit job for date/time
    – Else
        • Print that there was an error
        • Write out error log file
    – End if

• End loop
```

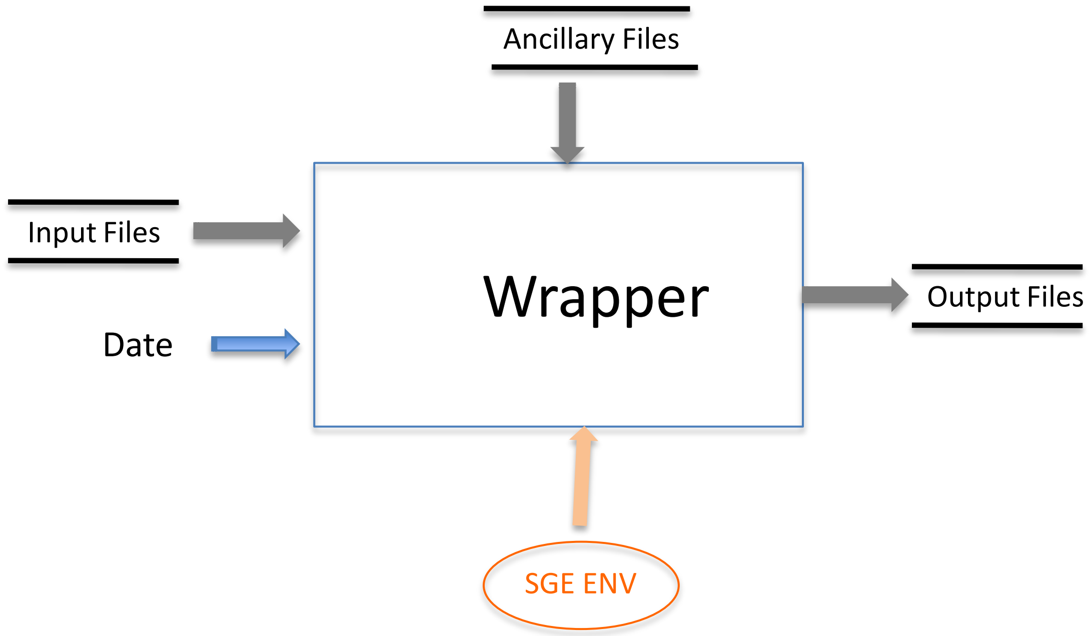
Submit Script Data Flow



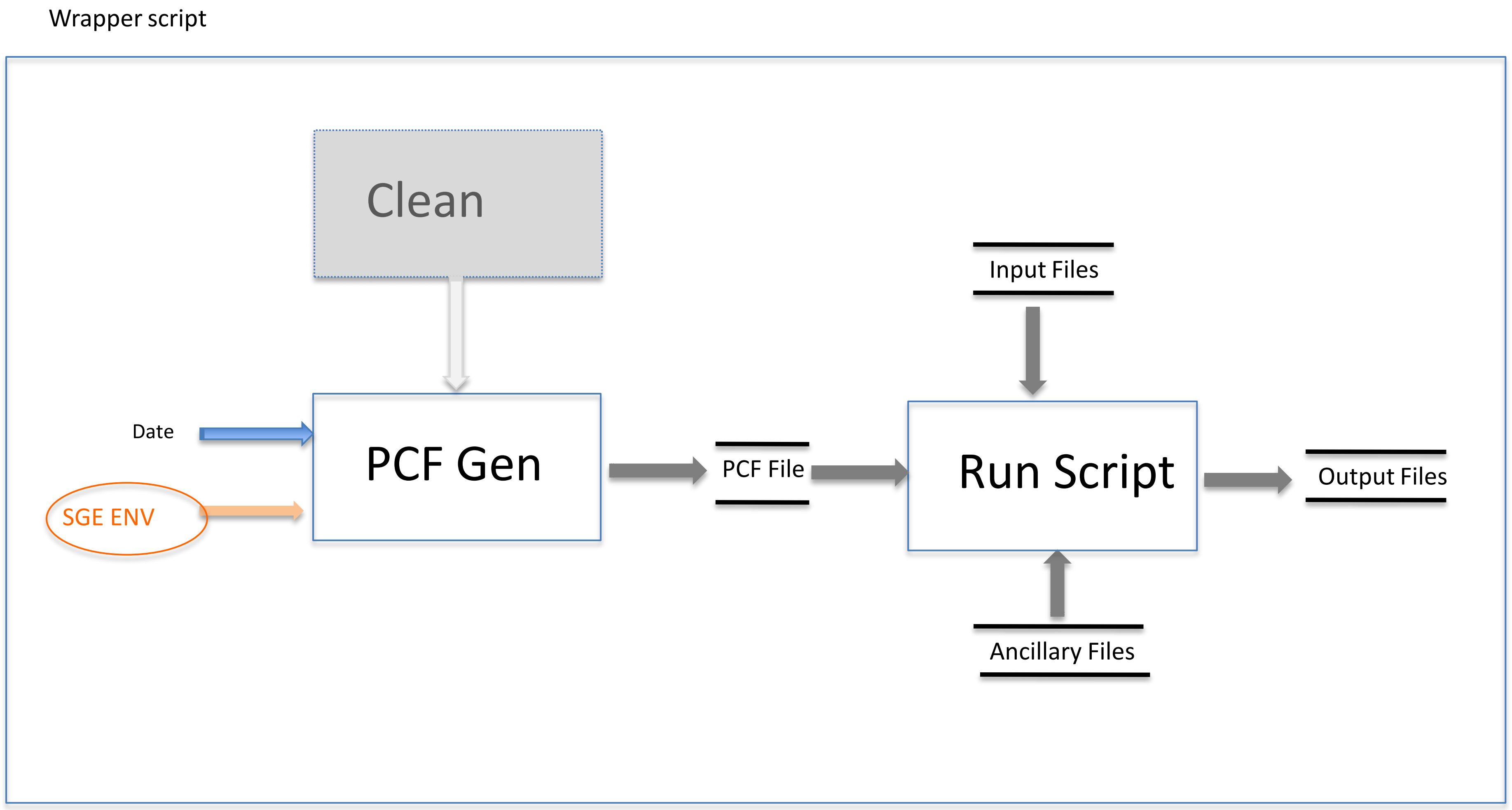
Submit Script Pseudo Code

- Read date/time from command line
- Create list of command line arguments for SGE’s qsub command
- Create command string from list for Wrapper script
- Put Wrapper script invocation on SGE queue (qsub)

Wrapper Script Data Flow



Detailed Wrapper Script Data Flow



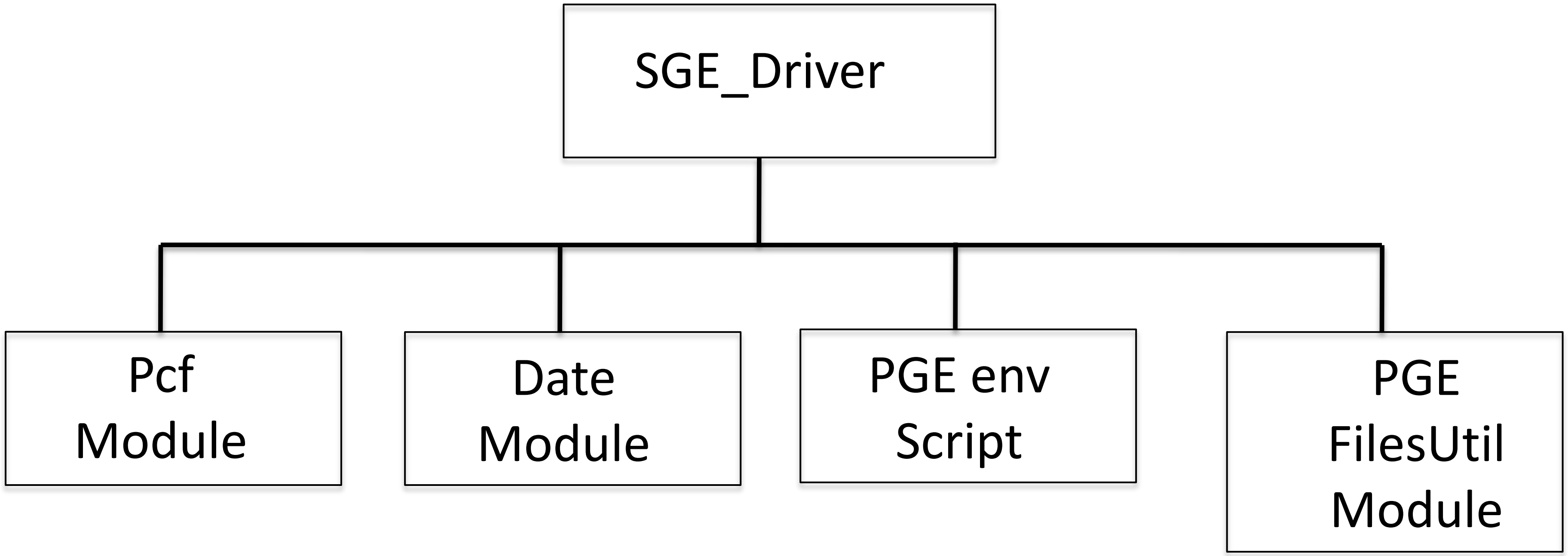
Wrapper Script Pseudo Code

- Read date/time from command line
- Copy all Vault hash variables into script’s ENV hash
- Set PGS include and message directories to be compatible with session’s platform
- Possibly Run Clean Script (TBD)
- Run PGE’s PCF generator script
- Execute PGE’s run script

Appendix C: Hierarchy Charts

- Encompasses code that is actually called or included by a script
- If the functions used vary with PGE, only their module will be listed
- Scripts called by Perl’s system command will NOT use shown
  - System launches a new session environment separate from parent script
  - Essentially program control is passed by this action until the subprocess completes.
- Names of listed here for Subsystem, PGE, SGE_Driver, Summit, and Wrapper are generic. Actual scripts would have the prefix of the PGE (i.e., CER9.4P2) or subsystem (i.e., instrument)

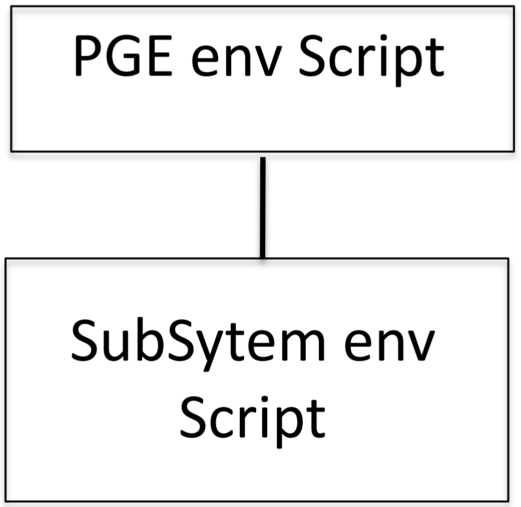
SGE_Driver Hierarchy Chart



SGE_Driver Notes

- Pcf module functions are used to print information on files to a log if mandatory files were not found.
- Dates module functions are used for checking date and time validity, breaking dates into components, and creating ranges of dates.
- PGE env Script populates subsystem hash. This hash is used as an input to the log file if processing cannot occur.
- PGE FileUtils modules contain all of the functions that check whether or not input files used by PGE exist.

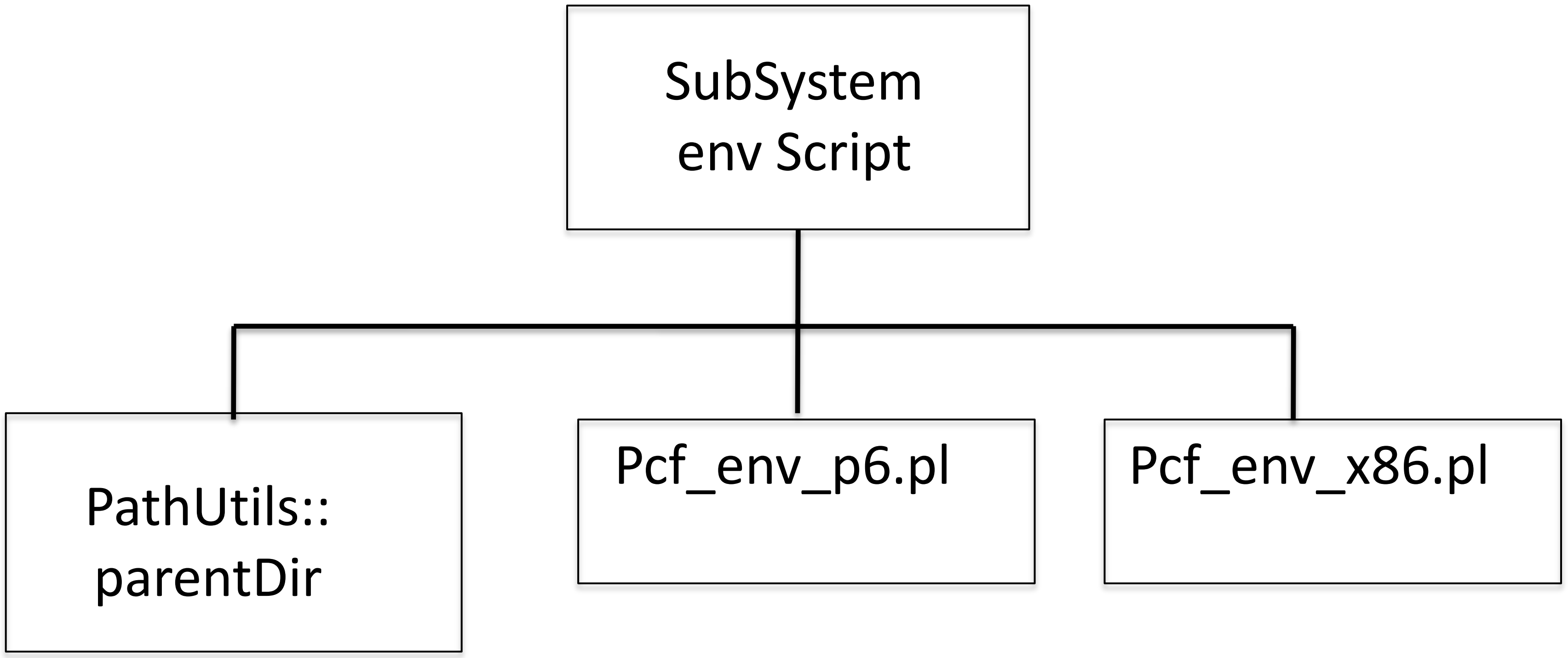
PGE env Script Hierarchy Chart



PGE env Script Notes

- Sets names of PGE specific directories in subsystem hash
- Sets values for variables used to create sampling strategy, production strategy, and configuration codes.
- Basically includes all of the code in the SubSystem env script. Thus when the PGE env script runs, it also runs the SubSystem env hash.

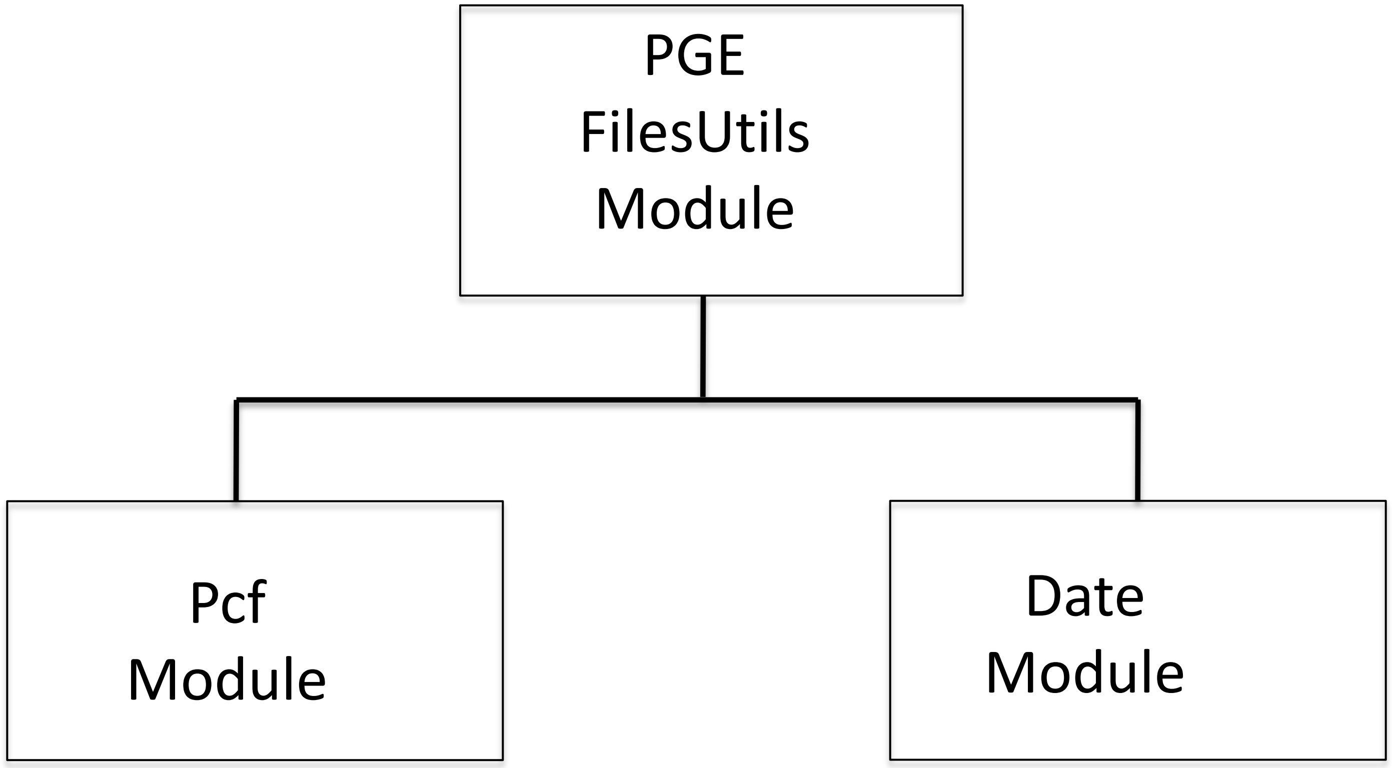
SubSystem env Script Hierarchy Chart



SubSystem env Script Notes

- Sets variables in subsystem hash for SubSystem directories. (home, run logs, ancillary, etc.)
- Use PathUtils::parentDir to determine home and CERES home directories
- Depending upon platform, includes pcf_env_p6.pl or pcf_env_x86.pl code. This means that it will perform all of these scripts code before running its own code. (pcf_env_p6.pl and pcf_env_x86.pl entries for all of the Toolkit variables into a hash.)

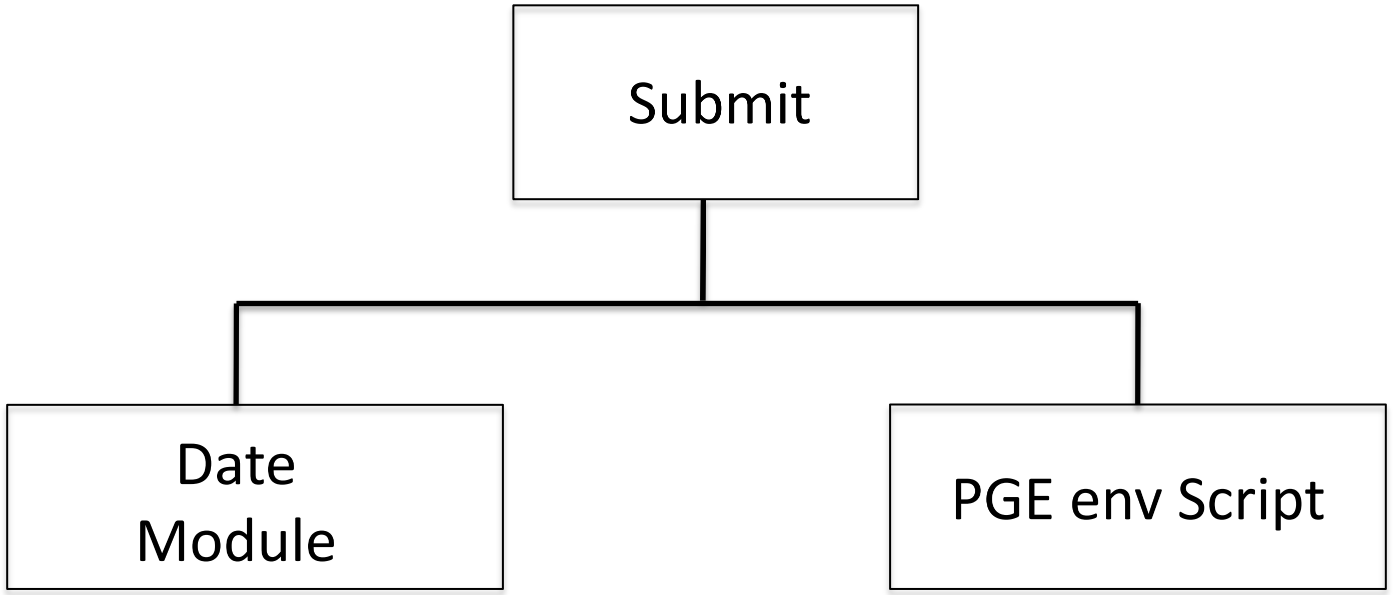
PGE FileUtils Module Hierarchy Chart



PGE FileUtils Script Notes

- Contains functions for creating input file names, searching for these files, and determining if all of the necessary files exist for processing.
- Typically has a different function for each type of input file.
- Uses Dates module function to manipulate date components and to generate a range of times or dates if need be.
- Use Pcf module functions to copy results to used or missing arrays internal to Pcf or, in some cases, to SGE_Driver existance.

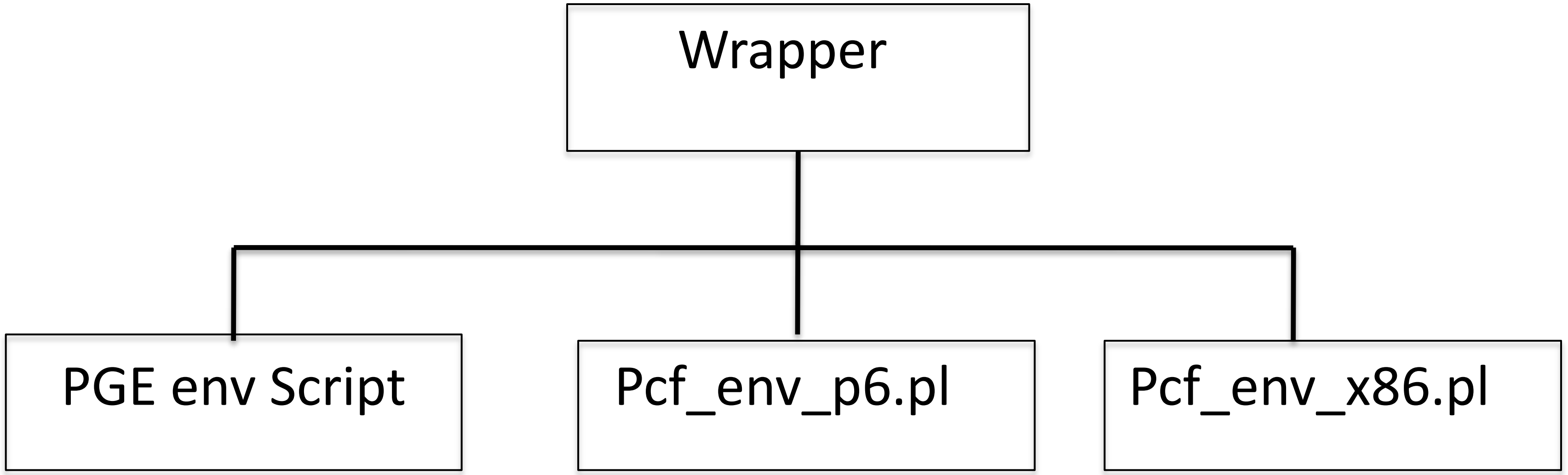
Submit Script Hierarchy Chart



Submit Script Notes

- Checks that date/time is valid
- Creates command line for SGE’s qsub command and then submits the corresponding wrapper script.
- Uses Dates module functions to check the validity of the date or time.
- Uses PGE’s environment script to setup the SGE log’s directory and to get the sampling strategy, processing strategy, and configuration code information from environment variables.

Wrapper Script Hierarchy Chart



Wrapper Script Notes

- Sets up Toolkit environmental variables then runs the PGE’s PCF generator followed by the corresponding run script.
- In some PGE’s a clean up script is run prior to the PCF generator script.
- Depending upon platform include either Pcf_env_p6.pl or Pcf_env_x86.pl. These contain Toolkit information.
- Uses PGE env script to get sampling strategy, production strategy, and configuration code from environmental variables. These are used to determine the PCF file name, which generally required as an input to the run script.